Schlüsselverwaltung im Sicheren Multicast

Abdulhadi Shoufan und Sorin A. Huss

Integrierte Schaltungen und Systeme
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
shoufan|huss@iss.tu-darmstadt.de

Abstract: In diesem Beitrag wird zunächst die Technologie des sicheren Multicast vorgestellt. Anschließend wird auf zwei wichtige Probleme der Schlüsselverwaltung im sicheren Multicast eingegangen: Die Skalierbarkeit von Rekeyinglösungen und die Evaluierung von Rekeyingalgorithmen. Zwei Lösungen, nämlich ein *Rekeying Prozessor* und ein *Rekeying Benchmark,* werden zur Behandlung dieser Probleme präsentiert.

1 Einführung

Nach einer relativ langen Zeit der Betrachtung als ein rein akademisches Problem gewinnt die Multicasttechnologie mit dem wachsenden Interesse am Internetfernsehen IPTV wieder an praktischer Bedeutung. Der IPTV Standard basiert nämlich auf dem IP-Multicast Protokoll. Laut aktuellen Markstudien [Ac07] stellen die Inhaltsrechte das am meisten diskutierte Thema im IPTV. Dies stellt hohe Anforderungen an die Multicasttechnologie, die ursprünglich als eine recht offene Technik bezüglich der Benutzerregistrierung gilt. Jeder Internetbenutzer kann sich jeder Multicastgruppe anschließen, ohne dass der Gruppenbesitzer, in diesem Fall der IPTV Provider, dies überhaupt erfahren muss. Um dennoch die Zugriffsrechte im Sinne eines kommerziellen Providers zu regeln, ist eine Verschlüsselung des Inhalts unerlässlich.

Diese Vorgehensweise wirft allerdings neue Probleme bezüglich der Schlüsselverwaltung auf, die für die Multicasttechnologie charakteristisch sind. Mit dem Bestreben der IPTV Provider, größte Benutzergruppen zu erreichen und hierbei einen flexiblen Dienst zu gewährleisten, so dass die An- und Abmeldung in die bzw. aus der Gruppe dynamisch bleibt, ergibt sich die folgende zentrale Anforderung: Es bedarf eines effizienten Schlüsselverwaltungsmechanismus, der den Gruppenschlüssel nach jeder Änderung in der Gruppenzusammensetzung aktualisiert, verschlüsselt und nur an die autorisierten Mitglieder versendet. Offensichtlich kostet dieser sogenannte Rekeyingprozess einen hohen Rechenaufwand auf der Serverseite und eine hohe Bandbreite auf dem Übertragungsnetz: ein Skalierbarkeitsproblem entsteht.

In der Literatur wurden zahlreiche Architekturen, Algorithmen und Protokolle publiziert, die dieses Skalierbarkeitsproblem adressieren. Lösungen zur Optimierung der Rekeyingperformanz konzentrieren sich auf die Minimierung der Anzahl der erforderlichen kryp-

tographischen Operationen und somit auf die Optimierung der Länge der Rekeyingnachrichten.

Aufgrund der Neuigkeit und der Komplexität dieses Forschungsgebiets für praktische Anwendungen lassen die präsentierten Lösungen eine einheitliche Methode zur Abschätzung der Rekeyingperformanz vermissen. In den meisten Fällen wird dadurch eine Evaluierung und ein direkter Vergleich von verschiedenen Rekeyingalgorithmen enorm erschwert.

Der vorliegende Beitrag erörtert zwei Hauptprobleme des Gruppenrekeying. Erstens wird eine innovative HW/SW-Architektur zur Optimierung der Rekeyingperformanz präsentiert. Im Unterschied zu bisherigen Lösungen wird die Rekeyingperformanz durch diese Architektur nicht nur auf der algorithmischen Rekeyingebene optimiert, sondern sowohl auf der kyptographischen Ebene als auch auf der Plattformebene. Diese neue Architektur wird als *Rekeying Prozessor* bezeichnet. Zweitens wird ein *Rekeying Benchmark* eingeführt, der einen einheitlichen und zuverlässigen Weg zur Abschätzung der Rekeyingperformanz verschiedener Rekeyingalgorithmen dargestellt.

Aufgrund der Tatsache, dass das Thema „sicheres Multicast“ noch keine generelle Bekanntheit genießt, wird Abschnitt 2 einer allgemeinen Einführung hierin gewidmet. Abschnitt 3 und Abschnitt 4 präsentieren dann den Rekeying Prozessor bzw. den Rekeying Benchmark.

2 Sicheres Multicast

Die Anwendung von Sicherheitsaspekten auf die Multicastkommunikation wirft spezielle Probleme auf, die in dem herkömmlichen Kommunikationsmodus eins-zu-eins unbekannt sind. Hierzu gehören vor allem die folgenden Gesichtpunkte:

1. Im sicheren Multicast basiert die Kommunikation auf einem Geheimnis, dem Gruppenschlüssel, der von einer großen Benutzergruppe geteilt ist. Je mehr Benutzer einen Schlüssel teilen, umso höher ist die Wahrscheinlichkeit, diesen Schlüssel zu enthüllen.
2. Die Sicherheit des Gruppenschlüssels hängt nicht nur von der Skalierbarkeit der Gruppe, sondern auch von ihrer Dynamik ab. So kann ein Internetbenutzer, der gestern als autorisiertes Gruppenmitglied galt, heute ein Sicherheitsrisiko darstellen.
3. Sicheres Multicast erfordert aufwändige Authentifizierungsmechanismen. In der herkömmlichen Unicasttechnologie können Nachrichtenauthentifizierungscodes (engl.: message authentication code, MAC) eingesetzt werden, um sicher zu stellen, dass die Nachricht von dem Sender stammt, der in der Nachricht angegeben ist. Da diese Verfahren auf einem vertraulichen Schlüssel zwischen dem Sender und dem Empfänger basieren, können sie für Multicast nicht ohne weiteres verwendet werden. Mit MAC-basierten Verfahren kann ein Gruppenmitglied nämlich lediglich feststellen, dass die Nachricht von innerhalb der Gruppe stammt (Gruppenauthentifizierung). Die eigentliche Identität des Senders ist hierbei jedoch nicht nachweisbar

(Quellenauthentifizierung). Um dies zu ermöglichen, müssen teure Signaturverfahren eingesetzt werden, die ursprünglich der höheren Sicherheitsanforderung der Nicht-Abstreitbarkeit dienen.

4. Der sicheren Kommunikation unterliegt stets eine Sicherheitsstrategie, die zwischen den Kommunikationspartnern verhandelt wird. Je mehr Benutzer eine sichere Kommunikation anstreben, umso schwieriger wird es, eine gemeinsame Sicherheitsstrategie zu definieren, die die unterschiedlichen Interessen aller Mitglieder vertritt.

Diese and andere Problematiken der sicheren Multicasttechnologie sind durch die Industrie und durch manche Forschungsinstitutionen früh erkannt worden, was zur Einrichtung von zwei Arbeitsgruppen geführt hat: *Secure Multicast Group* (SMuG) innerhalb der *Internet Research Task Force* (IRTF) in 1998 [Sm98] und die Arbeitsgruppe *Multicast Security* (MSEC) innerhalb der *Internet Engineering Task Force* (IETF) in 2000 [Ms00]. Ein entscheidender Beitrag dieser Gruppen war die Identifizierung von drei Problembereichen im Rahmen des sicheren Multicast [Ha03]:

1. **Die Behandlung sicherer Multicastdaten:** In diesem Problembereich wird die Übertragung sicherer Multicastdaten einschließlich der Datenvertraulichkeit, Datenintegrität und Quellenauthentifizierung untersucht. Die Anwendbarkeit bzw. die Erweiterung verfügbarer Sicherheitsprotokolle auf die Multicasttechnik stellt einen wichtigen Aspekt in diesem Bereich dar. Ein Beispiel hierfür präsentiert die Entwicklung des Protokolls *Multicast Encapsulating Secure Payload* (MESP) als Erweiterung der bekannten Unicast-Ausprägung (ESP), siehe [Ca00]. Ein weiteres Beispiel bezieht sich auf die oben beschriebene Quellenauthentifizierung.

2. **Schlüsselverwaltung:** Dieser Problembereich beschäftigt sich mit der Generierung und Verteilung des Kommunikationsschlüssels sowohl während der Registrierung in statischen Gruppen als auch nach jeder Änderung in der Gruppenzusammensetzung in dynamischen Gruppen. Im letzten Fall wird von Gruppenrekeying gesprochen. Der Schwerpunkt des vorliegenden Beitrags bezieht sich auf dieses Problem und wird in den nächsten Abschnitten näher betrachtet.

3. **Sicherheitsstrategien für Multicast:** Im sicheren Multicast können Gruppenmitglieder verschiedene Interessen und Verantwortungen vertreten. Eine Aufgabe der Sicherheitsstrategie besteht in der Rollenverteilung auf die Benutzer, beispielsweise als Sender, Empfänger, Gruppenbesitzer und Schlüsselserver. Dieser Problembereich beschäftigt sich außerdem mit der Festlegung der zu verwendende Algorithmen für Verschlüsselung, Authentifizierung, usw. Ist eine Strategievereinbarung praktisch nicht vertretbar, so sind Methoden zur Strategieerzwingung einzusetzen [Di00].

3 Rekeying Prozessor

Gruppenrekeying wirft ein Skalierbarkeitsproblem auf, wie bereits in Abschnitt 1 erwähnt wurde. Die Abmeldung eines Mitgliedes aus einer Gruppe mit *n* Benutzern erfordert nämlich die Generierung eines neuen Gruppenkommunikationsschlüssels und dessen Verschlüsselung mit dem eigenen Schlüssel jedes der bleibenden Benutzer. D.h., *n-1*

Verschlüsselungen müssen durch den Server berechnet werden und *n-1* Rekeyingnachrichten sind somit über das Netzwerk zu übertragen. Um diesen Aufwand zu reduzieren ist in der Literatur eine logische Schlüsselhierarchie (engl.: logical key hierarchy, LKH) vorgeschlagen worden, die die Rekeyingkosten derart reduziert, dass diese mit der Gruppengröße in einem logarithmischen Verhältnis stehen [Wo00]. Dieser Ansatz wird im nächsten Abschnitt beschrieben.

3.1 Logische Schlüsselhierarchie (LKH)

Die Grundidee des LKH Verfahrens besteht in der Aufteilung der Benutzer in hierarchische Untergruppen und der Zuweisung eines Hilfsschlüssels an jede Teilgruppe. Abbildung 1 (links) zeigt eine Gruppe aus 8 Benutzern. Die Mitglieder m_0 und m_1 bilden eine Untergruppe mit dem Hilfsschlüssel k_{0-1}. Mitglieder m_0, m_1, m_2, und m_3 gehören einer größeren Untergruppe an, der der Hilfsschlüssel k_{0-3} zugeordnet ist. Alle Mitglieder repräsentieren die größte Teilgruppe, deren Hilfsschlüssel gleich dem Kommunikationsschlüssel ist, d. h. $k_{0-7} = k_g$. Dementsprechend besitzt ein Benutzer in LKH mehrere Schlüssel: den eigenen Schlüssel k_d, der ausschließlich diesem Benutzer und dem Server bekannt ist, einige Hilfsschlüssel k_{x-y} und den Gruppenschlüssel k_g. m_6 besitzt beispielsweise $k_d = k_6$, $k_g = k_{0-7}$ und die zwei Hilfsschlüssel k_{6-7} und k_{4-7}.

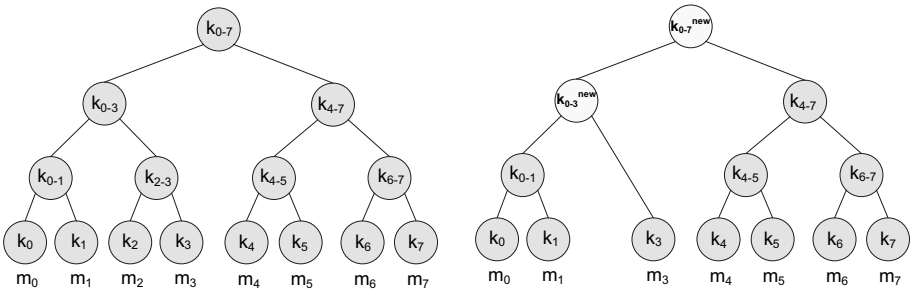

Abbildung 1: Logische Schlüsselhierarchie (LKH)

Die Funktionsweise des LKH Verfahrens wird an einem Beispiel erläutert. Eine Abmeldung des Benutzers m_2 erfordert die Löschung des Schlüssels k_{2-3} und die Aktualisierung der Schlüssel k_{0-3} und k_{0-7}, so dass der Schlüsselbaum nach diesem Vorgang wie im rechten Teil von Abbildung 1 erscheint. Die aktualisierten Schlüssel k_{0-3}^{new} und k_{0-7}^{new} müssen nun verteilt werden. Hierzu werden sie jeweils mit den Schlüsseln in den Sohnknoten verschlüsselt. Insgesamt werden die folgenden Rekeyingteilnachrichten generiert, wobei die Notation $E_{k_a}(k_b)$ die Verschlüsselung des Schlüssels k_b mit dem Schlüssel k_a repräsentiert.

$$E_{k_3}(k_{0-3}^{new}),\ E_{k_{0-1}}(k_{0-3}^{new}),\ E_{k_{0-3}^{new}}(k_{0-7}^{new}),\ E_{k_{4-7}}(k_{0-7}^{new})$$

Ohne LKH müsste der Server den Schlüssel k_{0-7}^{new} mit dem eigenen Schlüssel jedes bleibenden Mitglieds verschlüsseln, was einem Aufwand von 7 Verschlüsselungen entspricht. Der Vorteil von LKH wird umso ersichtlicher, je größer die Gruppe wird.

3.2 Motivation zum Rekeying Prozessor

Rekeyingnachrichten stellen äußerst kritische Daten dar, deren Manipulation das komplette System zum Stillstand bringen kann. Dementsprechend ist eine Verifizierung sowohl der Integrität dieser Daten als auch des eigentlichen Senders in den meisten Fällen unerlässlich. Wie in Abschnitt 2 erläutert wurde, erfordert dies den Einsatz digitaler Signaturen. Unter diesen Gegebenheiten ist ersichtlich, dass die Reduzierung der Anzahl der symmetrischen Verschlüsselungen nach LKH eine notwendige, jedoch keineswegs ausreichende Maßnahme zur Optimierung des Rechenaufwandes ist. Hierzu muss nämlich insbesondere das Signierverfahren in Betracht gezogen werden. Eine anerkannte Strategie zur Optimierung der Berechnung von digitalen Signaturen basiert auf der Hardwareunterstützung.

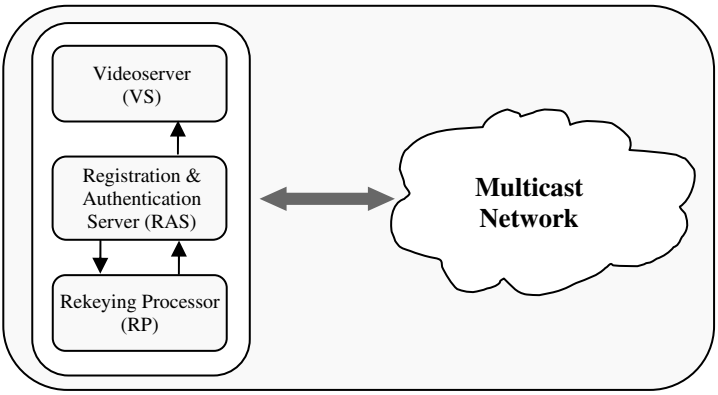

Abbildung 2: Pay-TV Szenario als Einsatzumgebung für den Rekeying Prozessor

Der in diesem Beitrag vorgestellten Rekeying Prozessor erzielt ein effizientes Gruppenrekeying nicht nur durch Reduzierung der Anzahl der notwendigen kryptographischen Operationen auf der Basis von LKH, sondern auch durch die Optimierung ihrer Ausführungszeiten. Hierbei zeichnet sich der Prozessor aufgrund seiner Hardware/Software Architektur durch einen hohen Grad an Flexibilität aus. Während die rechenintensiven Operationen, wie die Schlüsselgenerierung, die symmetrische Verschlüsselung, die Hashwertberechnung und die Signaturberechnung durch dedizierte Hardwaremodule effizient durchgeführt werden, übernimmt ein eingebetteter PowerPC Prozessor die Aufgabe der Schlüsselbaumverwaltung. Die gesamte Architektur ist auf einer FPGA Plattform realisiert, die als Koprozessor in einer Serverumgebung agiert. Abbildung 2 zeigt ein Multicast Pay-TV System als ein potentielles Szenario für die Anwendung des Rekeying Prozessors. Ein Videoserver (VS) verschlüsselt Videopakete mit einem Gruppenschlüssel k_g. Interessierte Internetbenutzer können sich beim Registrierungs- und Authentifizierungsserver (RAS) anmelden und erhalten daraufhin ihre Schlüssel. Der Rekeying Prozessor (RP) verwaltet den Schlüsselbaum und führt Anfragen des RAS in Form von Befehlen aus.

3.3 Architektur des Rekeying Prozessors

Abbildung 3 veranschaulicht die detaillierte Architektur des Rekeying Prozessors. Die Rekeyinganfragen werden durch den RAS in den Instruction-FIFO geschrieben. Der Rekeying Prozessor führt diese Befehle aus und schreibt die Rekeyingnachrichten in den Output-FIFO, die dann durch den RAS an die Gruppenbenutzer per Multicast weitergeleitet werden.

Der Rekeying Prozessor übernimmt die komplette Aufgabe der Baumverwaltung mit Hilfe des PowerPC Prozessors, der im FPGA XilinxII-Pro eingebettet ist. Der Schlüsselbaum wird in einen externen SDRAM auf der Hardwarekarte gespeichert. Für die Baumverwaltung unterstützt der Rekeying Prozessor zwei Modi: einen dynamischen und einen statischen Modus. Die dynamische Baumverwaltung basiert auf dynamischen Datenstrukturen, wie sie aus der Softwaretechnik bekannt sind. Die zusätzlichen Baumdaten, d. h. die Zeiger auf Vor- und Nachfolgerkonten und die Informationen über tiefsten und flachsten Blätter, werden mit den Nutzdaten, d. h. mit den Schlüsseln, im externen Speicher abgelegt. Die statische Baumverwaltung ist dagegen charakteristisch für den Rekeying Prozessor und basiert auf der Tatsache, dass die Dedizierung der Hardware für die Rekeyingaufgabe es erlaubt, einen statischen Speicherbereich vorzureservieren. Wird dieser Speicherbereich entsprechend der Rekeyingaufgabe organisiert, so kann die physikalische Topologie des Speichers so genutzt werden, dass eine effiziente Baumverwaltung sowohl bezüglich der Traversierungszeit als auch des Ressourcenbedarfs erzielt werden kann [Sh07]. Der optionale Speicher KSM wird für diesen Zweck verwendet.

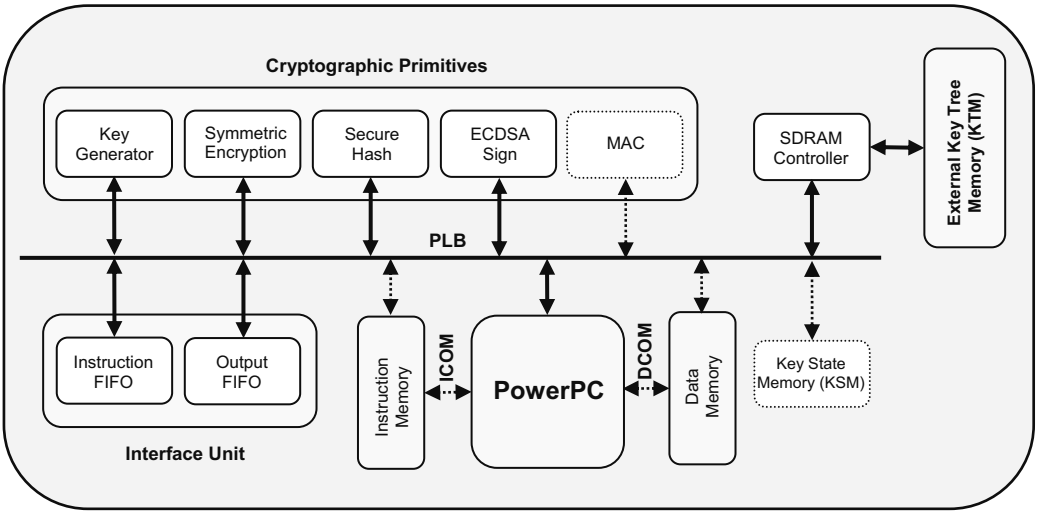

Abbildung 3: Architektur des Rekeying Prozessors

Zusätzlich zu der Vertraulichkeit und Integrität der Rekeyingdaten, die durch die Module Symmetric Encryption bzw. Secure Hash realisiert sind, bietet der Rekeying Prozessor eine Quellenauthentifikation durch das Modul ECDSA Sign und wahlweise Gruppe-

nauthentifikation durch das Modul MAC an. Alle weiteren Ausführungen in diesem Beitrag beschränken sich der Kürze halber auf die härtere Authentifikationsalternative, d. h. auf die Authentifikation mit digitalen Signaturen. Entsprechend dieser Architektur werden dem Rekeying Prozessor die folgenden Eigenschaften zugeschrieben:

1. *Hohe Performanz:* Aufgrund der Hardwareunterstützung ermöglicht der RP eine effiziente Benutzeranmeldung und -abmeldung in sehr großen Gruppen. Das direkte Resultat ist eine gesteigerte Dienstgüte bezüglich des Zugangs neuer Mitglieder und eine verbesserte Zugriffskontrolle bezüglich abgehender Benutzer. Der aktuelle Prototyp unterstützt eine Gruppengröße von bis zu 131.028 Benutzern.
2. *Hohe Angriffssicherheit:* Zusätzlich zu der erhöhten Sicherheit aufgrund des effizienten Rekeying bietet der Prozessor Schutz vor verschiedenen Angriffen auf den Server. Dies ist auf die Hardwareverwaltung der Schlüssel und darauf zurückzuführen, dass der RP alle Schlüssel ausschließlich in verschlüsselter Form liefert.
3. *Hohe Flexibilität:* Der RP zeichnet sich durch eine hohe Modularität bezüglich der Hardwarekomponenten aus. Dies erleichtert den Austausch und die Erweiterung der verschiedenen Module, um beispielsweise neuere kryptographische Algorithmen oder verschiedene Schlüssellängen zu unterstützen.

3.4 Rekeyingalgorithmen

Der Befehlsatz des Rekeying Prozessors besteht aus neun Instruktionen, wobei vier hiervon der Initialisierung der Sicherheitsmodule dienen. Drei weitere Befehle (*Join MEMID* k_d, *Disjoin MEMID* und *Resynchronize MEMID*) werden zur Anmeldung, Abmeldung bzw. Resynchronisation eines Benutzers ausgeführt, wobei MEMID die Identität eines Mitglieds darstellt. Zur Erhöhung der Sicherheit kann ein weiterer Befehl (*UpdateKg*) zur Aktualisierung des Gruppenschlüssels ausgeführt werden, falls die Gruppe über eine lange Zeitperiode unverändert bleibt. Ein letzter Befehl (*DeliverKg*) dient der Lieferung des Gruppenschlüssels an den Datenserver, z.B. den Videoserver. Zur Erläuterung stellt Algorithmus 1 die Ausführung des Befehls *Join* auf hohem Abstraktionsgrad dar. In Schritt 3 werden alle Schlüssel vom jeweiligen Blatt bis zur Wurzel generiert und verschlüsselt. Um nicht für jede Rekeyingteilnachricht eine digitale Signatur zu berechnen, wird ebenfalls in Schritt 3 der Hashwert h für alle Rekeyingteilnachrichten berechnet. In Schritt 4 wird dieser Wert abschließend signiert.

Algorithmus 1 Benutzeranmeldung

Eingang: MEMID, k_d
Ausgang: Rekeyingnachricht
1. Füge ein neues Blatt für ein Mitglied mit (MEMID, k_d) ein
2. Aktualisiere Baumtopologie
3. Aktualisiere die Schlüssel auf dem Anmeldepfad → h
4. Signiere h

Algorithmus 2 verdeutlicht das verwendete Signaturverfahren auf der Basis von elliptischen Kurven ECDSA (engl.: Elliptic Curve Digital Signature Algorithm) [Ie00]. n stellt hierbei die Ordnung der additiven Gruppe dar, die sich aus Punkten der elliptischen Kurve zusammensetzt, G ist der Generator dieser Gruppe. Im Schritt 2 wird eine Punkt-

multiplikation auf der Kurve durchgeführt. Dies ist die aufwändigste Operation der Signaturberechnung. Die Abszisse des resultierenden Punktes x wird in Schritt 3 auf r reduziert. Der Wert von r wird dann in Schritt 4 mit dem privaten Schlüssel des Servers d multipliziert. Das Ergebnis wird zum Hashwert addiert und das Ergebnis mit der invertierten Zufallszahl multipliziert. Die modulare Inversion von k in Schritt 4 stellt die zweitaufwändigste Operation dar.

Algorithmus 2 ECDSA

Eingang: Hashwert h
Ausgang: Digitale Signatur (r, s)
1. Wähle eine Zufallszahl k aus [1, n-1]
2. Bestimme $P(x, y) = kG$
3. Bestimme $r = x(mod\ n)$; wenn $r = 0$, zurück zu 1
4. Bestimme $s = k^{-1}(h + dr)(mod\ n)$; wenn $s = 0$, zurück zu 1

3.5 HW/SW Codesign und Ergebnisse

Die sequentielle Beschreibung der Rekeyingoperation entsprechend Algorithmus 1 und Algorithmus 2 ist nicht geeignet für eine HW/SW Realisierung. Um die Hardware- und Softwareressourcen des FPGA effizient auszunutzen, ist eine Notation erforderlich, die den inhärenten Parallelismus im gesamten Rekeyingalgorithmus wiedergibt. Solch eine Anforderung erfüllt ein Taskgraph als Algorithmusnotation. Abbildung 4 zeigt, wie der Rekeyingalgorithmus, nun repräsentiert als Ablaufdiagramm, auf einen Taskgraph abgebildet werden kann.

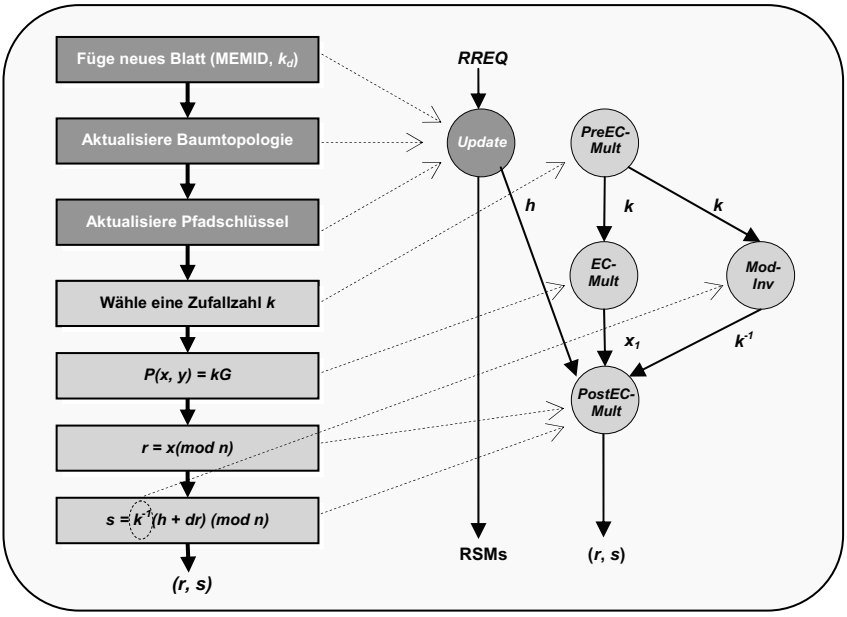

Abbildung 4: Extraktion eines Taskgraphs aus dem Rekeyingalgorithmus

Diese Abbildung basiert auf der bereits beschriebenen Erkenntnis, dass die Punktmultiplikation auf der elliptischen Kurve und die modulare Inversion die rechenintensivsten Operationen sind, und ihnen daher dedizierte Tasks im Taskgraph, hier *ECMult* und *ModInv* genannt, zugewiesen werden. Der Task *Update* umfasst alle Operationen der Baumverwaltung, Schlüsselgenerierung, Verschlüsselung und der Hashwertberechnung. Die gesamte Ausführungszeit dieser Operationen ist vergleichbar mit der Zeit der modularen Inversion, wenn sie in Software realisiert werden. Dies entspricht den Messungen anhand verschiedener Entwurfalternativen, von denen hier allerdings aus Platzgründen ausschließlich diejenige präsentiert wird, die die Entwurfsziele der Performanz, des Ressourcebedarfs und der Flexibilität optimal erfüllt. Abbildung 5 zeigt die Abbildung der verschiedenen Tasks auf die Hardware- und Softwareressourcen für diese Entwurfsalternative.

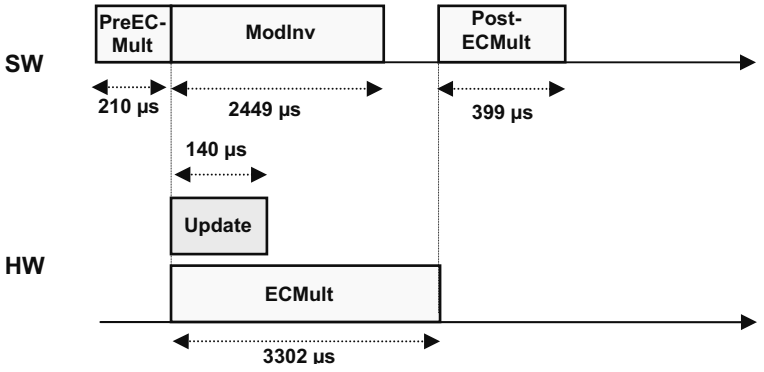

Abbildung 5: Bindung und Ablaufplanung der gewählten Entwurfalternative

Entsprechend Abbildung 5 dauert die Anmeldung eines neuen Mitglieds im ungünstigsten Fall 3.91 ms. Dieser Fall tritt bei der Zuordnung des neuen Mitglieds zu einem Blatt auf, so dass der längste Pfad zur Wurzel entsteht. Solch eine Situation kommt umso häufiger vor, je größer die Gruppe wird.

Tabelle 1: Performanzvergleich des RP mit anderen Vorarbeiten

		[Am04]	**[Wo00]**	**RP**
Kryptographische Primitive	**Verschlüsselung**	n/a	DES-CBC	AES-128
	Schlüsselgenerierung	n/a	n/a	ANSI X9.17
	HashFunktion	n/a	MD5	AES-basierte Meyer Hashfunktion
	Digitale Signatur	RSA-1024	RSA-512	ECDSA-128
Gruppengröße		50	8.192	131.072
Ausführungszeit		Up to 640 *ms*	12 *ms* – 16,2 *ms*	3,91 *ms*
Messbedingungen		Einschließlich Kommunikationskosten	Berechnungszeit des Servers (Mittelwert)	Berechnungszeit des RP (Worst case)

Um die inhärenten Vorteile des Rekeying Prozessors zu verdeutlichen präsentiert Tabelle 1 einen Vergleich mit zwei Lösungen aus der Literatur. Die Betrachtung der Ergebnisse aus [Am04] liegt darin begründet – trotz der komplexen Messbedingungen, dass diese Arbeit zu den seltenen Publikationen gehört, die eine prototypische Lösung mit umfassender Betrachtung der Sicherheitsanforderungen einschließlich der Quellenauthentifizierung beschreiben. Der nächste Abschnitt geht auf das Problem der Performanzevaluierung von Rekeyingalgorithmen ein und präsentiert ein Lösungsverfahren hierzu.

4 Rekeying Benchmark

Der Vergleich von Rekeyingalgorithmen stellt eine äußerst problematische Aufgabe dar. Dies liegt an den unterschiedlichsten Metriken, Parametern, und Methoden, die in der Literatur zur Bewertung der Rekeyingperformanz herangezogen werden. Dies wird in den folgenden drei Punkten näher erläutert:

1. Als Performanzmetriken kommen in publizierten Vorarbeiten unter anderem die Anzahl der Verschlüsselungen [Wo00, Li01], die Anzahl der Nachrichten pro Minute [Wa99], die Anzahl der aktualisierten Schlüssel [Ng05] und die Verarbeitungszeit einer Anfrage [Am04] in Betracht.
2. Als Parameter, in deren Abhängigkeit die Performanzmetriken gemessen werden, werden die Gruppengröße [Wo00, Ng05], die Anzahl der Anmelde-, und Abmeldeanfragen [Li01, Am04], die Beobachtungszeit [Pe03], das Rekeyingintervall [Ch02] und die Baumhöhe [Wo00, Sh03] in Erwägung gezogen.
3. Als Methoden zur Abschätzung der Rekeyingperformanz kann zwischen Analytischen Verfahren [Wo00, Sh03], simulationsbasierten Methoden [Ng05, Lu05] und Messungen an Prototypen [Am04, Mo99] unterschieden werden.

4.1 Zuverlässige Evaluierung von Rekeyingalgorithmen

Die oben geschilderte Situation hat zur Erschwerung oder gar zur Verhinderung einer objektiven Evaluierung von Rekeyingalgorithmen geführt. Die Einheitlichkeit in den verwendeten Metriken, Parametern und Methoden stellt hier allerdings nicht die einzige Voraussetzung für diese zuverlässige Evaluierung dar. Vielmehr handelt es sich um die Auswahl dieser Größen und Methoden, wie es im Folgenden vorgestellt wird [Sh07a].

1. *Auswahl der Evaluierungsmethode:* Analytische Methoden basieren stets auf vereinfachten Modellen und können ausschließlich für eine allgemeine Abschätzung verwendet werden. Dynamische Effekte können hierbei nur eingeschränkt berücksichtigt werden. Messungen an Prototypen sind stark von den verwendeten kryptographischen Algorithmen und von der Ausführungsplattform abhängig. Simulationsmethoden stellen trotz verminderter Performanz verglichen mit der letzten Methode die einzige Wahl dar, die eine intensive Untersuchung der Rekeyingkosten erlaubt, ohne hierbei von den darunter liegenden kryptographischen Operationen und von der Ausführungsplattform abhängig zu sein.

2. *Auswahl der Performanzmetriken:* Einer zuverlässigen Abschätzung der Rekeyingperformanz sollten Metriken zugrunde liegen, die aus der Systemsicht relevant sind. Hierbei hilft eine genaue Analyse der Systemanforderungen an einen Rekeyingalgorithmus. Das System erfordert einen Rekeyingalgorithmus, der es ermöglicht, Gruppenbenutzer rasch an- und abzumelden. Eine schnelle Anmeldung hat einen direkten Einfluss auf die Systemdienstgüte, da neue Benutzer nicht lange auf ihre Verbindung warten müssen. Eine schnelle Abmeldung erhöht die Systemzugriffskontrolle und dient somit der Systemsicherheit. Zwei entsprechende Metriken werden eingeführt: die *Rekeyingdienstgüte* (RQoS) und die *Rekeyingzugriffskontrolle* (RAC).
3. *Auswahl der Performanzparameter:* Sinnvollerweise sollen die Performanzmetriken in Abhängigkeit von Parametern abgeschätzt werden, die ebenfalls aus der Systemsicht von Bedeutung sind. In der Rekeyingaufgabe strebt das System möglichst hohe Werte für RQoS und RAC in Bezug auf die Gruppengröße und Gruppendynamik an. Daher bestimmt der Rekeying-Benchmark die Performanzmetriken als Funktionen der aktuellen Gruppengröße n, der An- und Abmelderaten λ bzw. μ.

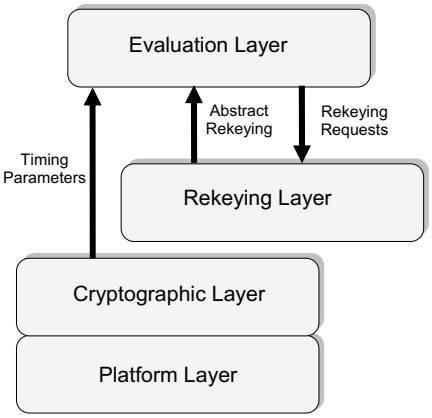

Abbildung 6: Abstraktionsmodell für den Rekeying Benchmark

Dementsprechend lässt sich das Konzept des vorgeschlagenen Benchmarks wie in Abbildung 6 gezeigt veranschaulichen. Die Trennung der Rekeyingschicht von den unteren Schichten dient dem oben erwähnten Ziel, die Rekeyingkosten unabhängig von den kryptographischen Algorithmen und von der Ausführungsplattform zu bestimmen. In der Benchmarkumgebung führen die zu evaluierenden Rekeyingalgorithmen somit überhaupt keine kryptographischen Operationen durch. Stattdessen bestimmen die Rekeyingalgorithmen die Anzahl der notwendigen Kryptoperationen (*Abtract Rekeying Costs*). Um die wirklichen Rekeyingkosten zu bestimmen multipliziert die Evaulierungsschicht diese abstrakten Kosten mit expliziten Zeitparametern (*Timing Parameter*), die letztlich die Ausführungszeiten der kryptographischen Operation repräsentieren. Ein Beispiel hierfür ist C_h, der die Ausführungszeit einer Hashwertberechnung darstellt. Da diese Parameter für die Evaluierung verschiedener Rekeyingalgorithmen einheitlich eingestellt werden können, liefert der vorgeschlagene Benchmark zuverlässige Ergebnisse.

4.2 Rekeying Benchmark: Architektur und Simulationsmodi

Abbildung 7 zeigt die allgemeine Architektur des Rekeying Benchmark mit zwei Schnittstellen und drei Komponenten. Die Benutzerschnittstelle *UI* ermöglicht die Bedienung des Benchmarks zur Evaluierung unterschiedlicher Algorithmen. Hierzu wählt der Benutzer die gewünschten Algorithmen und den Simulationsmodus aus.
Der Rekeying Benchmark unterstützt dabei unterschiedliche Simulationsmodi zur Messung der Rekeyingperformanz (RQoS und RAC) als Funktionen der Gruppengröße, der Anmelderate und der Abmelderate, siehe Abschnitt 4.1. Als Benutzerinterface können darüber hinaus verschiedene Systemparameter, Gruppenparameter und die oben beschriebenen Zeitparameter einstellt werden. Die Programmierschnittstelle *PI* wird unter anderem zur Einbindung neuer Algorithmen genutzt. Der *Request Generator* erzeugt entsprechend der Benutzereinstellungen eine Liste von Anfragen. Eine Anfrage wird durch drei Attribute beschrieben: den Typ (Anmelden, Abmelden), die Ankunftszeit der Anfrage und die Mitgliedsidentität. Die Anfragen werden dem *Algorithm Manager* übermittelt, der diese an alle zu simulierenden Algorithmen weiterleitet. Die resultierenden Rekeyingkosten werden schließlich durch den *Performance Evaluator* bewertet. Dieser bestimmt die Performanzmetriken und verbindet sie für die Anzeige mit den entsprechenden Parametern n, λ bzw. μ, je nach Simulationsart.

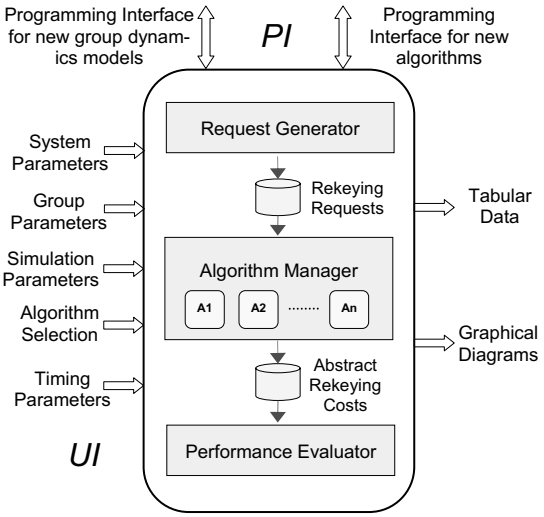

Abbildung 7: Softwarearchitektur des Rekeying Benchmark

5 Zusammenfassung

In diesem Beitrag wurden zwei Lösungen zu zentralen Problemen der Schlüsselverwaltung im sicheren Multicast präsentiert. Eine hocheffiziente Lösung zum Skalierbarkeitsproblem, der Rekeying Prozessor, wurde vorgeschlagen und eine zuverlässige Evaluierungsumgebung für unterschiedliche Rekeyingalgorithmen ist vorgestellt worden.

Literaturverzeichnis

[Ac07] Accenture Ltd. „IPTV Monitor-Issue 4“, www.accenture.com. April 2007.

[Am04] Amir Y., Kim Y., Nita-Rotaru C., and Tsudik G., “On the Performance of Group Key Agreement Protocols”, ACM Trans. on Information Systems Security, 7(3), 2004, pp. 457-488.

[Ca00] Canetti R., Rohatgi P., and Cheng P., “Multicast Data Security Transformations: Requirements, Considerations, and Protocol Design”, draft-irtf-smug-data-trasforms-00.txt, IRTF, June 2000.

[Ch02] Chen W., and Dondeti L.R., “Performance Comparison of Stateful and Stateless Group Rekeying Algorithms”, in Proc. of Int. Workshop in Networked Group Communication, 2002.

[Di00] Dinsmore P.T., et al., “Policy-based Security Management for Large Dynamic Groups: an Overview of the DCCM Project”, in Proc. of the DAPRA Information Survivability Conference and Exposition, Vol. I, Hilton Head, pp. 64-73, January 2000.

[Ha03] Hardjono T., and Dondeti L.R., “Multicast and Group Security”, Artech House, 2003.

[Ie00] IEEE (Institute of Electrical and Electronics Engineers), “Standard Specifications for Public-Key Cryptography”, Annex A, 2000. http://grouper.ieee.org/groups/1361/.

[Li01] Li X., Yang Y.R., Gouda M., Lam S.S., “Batch rekeying for Secure Group communications”, in Proc. of Int. WWW Conf., Hong Kong, May 2001.

[Lu05] Lu H., “ A Novel High-Order tree for Secure Multicast Key Management”, IEEE Trans. on Computers, vol. 54, No. 2, February 2005, p. 214-224.

[Mo99] Moyer M.J., Tech G., Rao J.R., and Rohatgi P., “Maintaining Balanced Key Trees for Secure Multicast”, Internet draft, June, 1999, http://www.securemulticast.org/draft-irtf-smug-key-tree-balance-00.txt.

[Ms00] http://www.ietf.org/html.charters/msec-charter.html.

[Ng08] Ng W.H.D., and Sun Z., “Multi-Layers Balanced LKH”, in Proc. of IEEE Int. Conf. on Communication ICC, May 2005, pp. 1015-1019.

[Pe03] Pegueroles, and Rico-Novella F., “Balanced Batch LKH: New Proposal, Implementation and Performance Evaluation”, in Proc. IEEE Symp. on Comupters and Communications, 2003, pp. 815.

[Sh03] Sherman A.T., et al., “Key Establishment in Large Dynamic Groups Using One-Way Function Trees”, IEEE Trans. on Software Engineering, Vol. 29, No. 5, pp. 444-458, May 2003.

[Sh07] Shoufan, A. “High Performance Group Key Management: A Way to scalable Internet Television”, VDM Verlag Dr. Müller, ISBN: 978-3-8364-2128-7, 2007.

[Sh07a] Shoufan A., Laue R., and Huss S.A., “Reliable Performance Evaluation of Rekeying Algorithms in Secure Multicast”, IEEE Int. Symposium on a World of Wireless, Mobile and Multimedia Networks, Helsinki, Finland, June 2007.

[Sm98] http://www.securemulticast.org/smug-index.htm.

[Wa99] Waldvogel M., Caronni G., Sun D., Weiler N., and Plattner B., “The VersaKey Framework: Versatile Group Key Management”, IEEE J. on Selected Areas in Communications, Vol. 17, No. 8, august 1999, pp. 1614-1631.

[Wo00] Wong C.K., Gouda M., Lam S.S., “Secure Group Communications Using Key Graphs”, IEEE/ACM Trans. on Networking, Vol. 8, No. 1, pp. 16-30, February 2000.